Foiling Sybils with HAPS in Permissionless Systems: An Address-based Peer Sampling Service

Amaury Bouchra Pilet, Davide Frey, François Taïani

Foiling Sybils with HAPS in Permissionless Systems: An Address-based Peer Sampling Service

Amaury Bouchra Pilet
École Normale Supérieure « Ulm »
Univ Rennes, Inria, CNRS, IRISA
Paris/Rennes, France

Davide Frey
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

François Taïani
Univ Rennes, Inria, CNRS, IRISA
Rennes, France

This paper has been published in the 2020 IEEE Symposium on Computers and Communications (ISCC), its published version is available on IEEE Xplore.
DOI: 10.1109/ISCC50000.2020.9219606
Link: https://ieeexplore.ieee.org/document/9219606

***Abstract*—Blockchains and distributed ledgers have brought renewed interest in Byzantine fault-tolerant protocols and decentralized systems, two domains studied for several decades. Recent promising works have in particular proposed to use epidemic protocols to overcome the limitations of popular Blockchain mechanisms, such as proof-of-stake or proof-of-work. These works unfortunately assume a perfect peer-sampling service, immune to malicious attacks, a property that is difficult and costly to achieve. We revisit this fundamental problem in this paper, and propose a novel Byzantine-tolerant peer-sampling service that is resilient to Sybil attacks in open systems by exploiting the underlying structure of wide-area networks.**

I. Introduction

Blockchains [1] have had a profound impact on both academia and industry over the last decade. They have introduced seminal mechanisms, such as proof-of-work (PoW) [1] and proof-of-stake (PoS) [2], which harden open (a.k.a. permissionless) systems against *Sybil attacks* [3], i.e. the possibility for malicious peers to generate many identities and hijack the system from honest peers.

Proof-of-Work (PoW) is unfortunately extremely costly, whereas Proof-of-Stake strongly links a peer's influence to its wealth (or stake), a problematic dependence in many applications. These limitations have led to alternative mechanisms against Sybil attacks, which exploit epidemic metastable phenomena [4] and unbiased sampling to test the likelihood of specific global predicates [5]. This idea has since been extended to the implementation of efficient Byzantine-resilient primitives in permission-less systems [6]. These promising techniques assume, however, a perfect *Random Peer Sampling* (RPS) service [7], [8], that is immune to Sybil attacks.

Unfortunately, existing Byzantine-tolerant RPS protocols are typically not applicable to open systems. They largely ignore Sybil attacks, and work instead in closed (a.k.a. permissioned) systems, in which participation in the system is controlled by a central authority [9], [10].

In this paper, we overcome this fundamental limitation, and propose HAPS, a novel Sybil-tolerant Random Peer Sampling service that exploit the hierarchical nature of the addressing schemes used on networks like the Internet to prevent attackers from flooding honest peers with references to malicious peers.

Our experimental evaluation shows that while HAPS has a cost in term of performance, its impact remains acceptable. In particular we show that the convergence speed of a privacy-preserving averaging algorithm (a typical use of an RPS protocol) executing on top of our solution remains in the same order of magnitude as with an unprotected peer sampling protocol.

II. System Model and Background

We consider an asynchronous decentralized system in which Peers communicate with each-other by means of point-to-point messages that carry the sender's address. We assume recipients can verify whether the alleged sender is online using ping messages, and that peer addresses follow a hierarchical structure (as is the case for IP addresses).

Peer-Sampling Protocols. A peer sampling service [7] provides each peer with a continuously changing sample of the network, the *view*, from which to choose communication partners. Peers periodically exchange subsets of their views and mix them. After some iterations, extracting a peer from a view approximates extracting one randomly from the entire network [7], [9].

Attacker Model and Assumptions. We consider an attacker who controls several peers (referred to as *malicious peers* in the following). The attacker seeks to perform a *Sybil attack* with the purpose of isolating a target peer from the rest of the network. To do this, the attacker attempts to remove the target peer from the views of other non-malicious peers, and to fill the target peer's view with references to malicious peers. The attacker can implement the Sybil attack in (a combination of) two ways. He/she can advertise either the IP addresses of malicious peers under his/her control, or fake addresses that belong to no

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

one. Advertising addresses of non-malicious peers provides no benefit to the attack.

In terms of attack power, we assume that the attacker can easily acquire a large number of addresses within the same, relatively small subnet. We also assume that acquiring addresses on multiple, very diverse subnets results in too high a cost for the attacker. This is because an attacker wishing to have very diverse addresses would need to buy an address in each or most existing subnets, or rely on botnets.

III. Hierarchically Addressed Peer Sampling

We introduce HAPS, a novel peer-sampling protocol designed to resist Sybil attacks in hierarchical networks under the assumption that network participants are uniformly distributed across the network's address space. As in other peer-sampling protocols, HAPS peers maintain a data structure, known as *view*, that contains references to other peers. Peers exchange messages containing subsets of these references. However, HAPS incorporates two important differences.

First, as in [9], peers only exchange their views in the *pull* phase of the protocol, i.e. when requesting information from another peer. Specifically, a peer periodically contacts a random neighbor and sends its own address and identifier to it. This message serves two purposes. It makes the peer known to the remote peer, and acts as a pull-request to which the remote peer responds with a random subset of its own view. This model decreases the risk of push-based pollution attacks [9].

Second, HAPS penalizes the choice of peers in overpopulated regions of the address space and associates equally sized portions of it with the same probability to be chosen, regardless of the number of peers they actually contain. This prevents an attacker from flooding a target peer's view with a large number of malicious peers from the same portion of the address space (subnet).

HAPS defines *equal-probability groups* as groups of addresses that share the same address prefix of length l_p. We refer to parameter l_p as the *equal-probability prefix length*. HAPS operates by giving each *equal-probability group* the same probability to be chosen when selecting a random peer. Thus peers in more populated groups end up having lower individual probabilities to be picked. As a result, if a peer's view gets polluted by a large number of peers from a unique attacker, with similar addresses, the probability for these peers to be selected randomly will remain low. This makes attacks relying on a large number of nodes from a small (relative to l_p) subnet completely ineffective.

The use of *equal-probability groups* forces HAPS to maintain larger sets of peers in its views to provide the same level of diversity as a standard peer-sampling protocol. Thus, while classical peer-sampling protocols maintain views of constant size, HAPS allows its views to grow for some time in order to populate each *equal-probability group* with a sufficiently large number of peers and uses two mechanisms for removing entries from views.

First, HAPS constantly checks for non-responding peers when exchanging information about them or attempting to contact them. In particular, it pings peers before sending references about them to other peers. If either of these operations detects a non-responding peer, HAPS removes it from its view.

Second, HAPS limits the size of its view by periodically removing peers from over-represented areas of the address space. To this end, HAPS defines a second important parameter, a *trimming prefix length*, l_t. A periodic *trimming* operation ensures that each view contains at most one peer for each prefix of length l_t.

To counter attackers that spoof their IP addresses, HAPS also pings peers before adding them to its view. Receiving an answer ensures that an address effectively belongs to a peer executing the protocol. This effectively prevents attacks based on the injection of fake addresses to isolate a peer. Injecting addresses of real peers not under his/her control provides no benefit to the attacker.

A. Buffered Probabilistic Binary Address Tree

HAPS implements *equal-probability groups* by storing references to peers in a novel data structure, the *Buffered Probabilistic Binary Address Tree* (*BPBA-Tree*), replacing the typical set used by other peer-sampling protocols. A BPBA-Tree consists of a binary tree with one leaf for each peer in the view and such that (i) the identifier of each leaf of the tree corresponds to the address of a peer; and (ii) the identifier a non-leaf node consists of the common prefix of the identifiers of all of its children. Figure 1 provides an example of a BPBA-Tree. For conciseness, we represent the prefixes corresponding to node identifiers as $hhhh/l$ where $hhhh$ is a hex representation on 16 bits, and l is a prefix length. For example, the notation 4000/2 corresponds to the binary prefix 01.

The two parameters of HAPS, l_p and l_t, identify specific levels in the BPBA-Tree and allow us to characterize some specific tree nodes. First, we define *equal-probability leaves* (EP-leaves) as the nodes whose identifier is at least as long as l_p and whose parent's identifier is strictly shorter than l_p. Clearly, a tree node with an identifier of length l_p is an equal-probability leaf (node 6380/10 in Figure 1). But more generally, an equal-probability leaf can have a longer identifier provided that its parent's identifier is shorter than l_p (all other equal-probability leaves in Figure 1). From the point of view of HAPS, each EP-leaf defines an *equal-probability group* consisting of the leaf nodes of the subtree that descends from it. We also define *trim leaves* (TR-leaves) as the nodes whose identifier is at least as long as l_t and whose parent's identifier is strictly shorter than l_t. Clearly EP- or TR-leaves need not be leaves of the BPBA-Tree. In Figure 1, we refer to the all descendants of EP-leaves as *random nodes*, and to all the descendants of TR-leaves as *temporary nodes* for reasons we clarify in

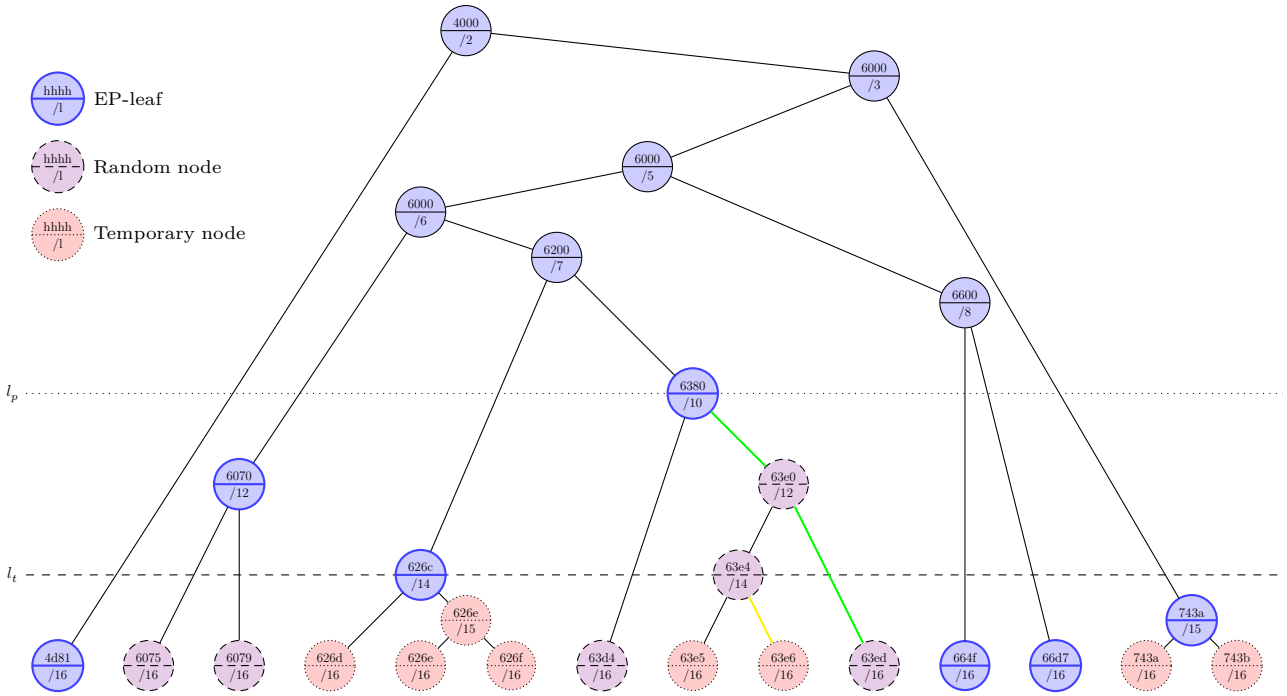


Figure 1: Example of Buffered Probabilistic Binary Addresses Tree

Section III-B. Since $l_p \leq l_t$, a temporary node is also a random node.

B. BPBA-Tree in HAPS

Peers use the BPBA-Tree as a container for their local views. The protocol thus uses the tree for four operations: inserting peer addresses into the tree, removing peer addresses, selecting a random peer, and periodically trimming the tree.

HAPS inserts a new peer as a tree leaf whenever it learns about it in push or pull operations. To do so, HAPS simply navigates from the root by selecting at each step the child whose identifier is a prefix of the new peer's address. It then adds the peer as a new leaf if it is not already present. Note that this also involves adding an intermediary internal node to accommodate the new leaf.

HAPS removes a peer from the tree whenever it detects it as non-responding, either because a send operation failed, or because the peer did not respond to a ping message. In either case, the protocol navigates from the root to the corresponding leaf node and removes it together with its parent internal node. The removed leaf's sibling connects to the removed internal node's parent.

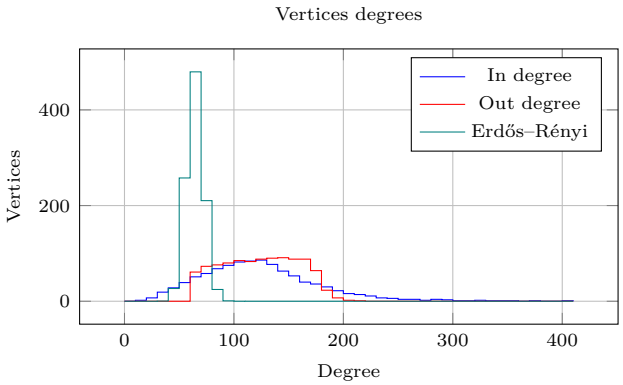
HAPS needs to select a random peer both when selecting a peer to communicate with when sending a request message, or whenever selecting a subset of its view to send when sending a pull message in response to a request. In both cases, HAPS selects each peer by first selecting an equal-probability leaf uniformly at random. For example, in Figure 1, it may select node 6380/10. In accordance with their name, all equal probability leaves thus have the same probability to be selected in this step. Then HAPS traverses the subtree rooted at the selected equal-probability leaf by randomly choosing one of the two child nodes (each with probability 0.5) at each step until reaching a leaf node, which represents the selected address. For example, HAPS may choose $6380/10 \rightarrow 63e0/12 \rightarrow 63ed/16$, with a $1/2^2 = 1/4$ probability.

As explained above, this selection process gives a lower weight to peers in highly populated subnets. The BPBA-Tree provides a level of diversity in peer selection that depends only on the *equal-probability prefix length*, being comparable to that of a view of size 2^{l_p}.

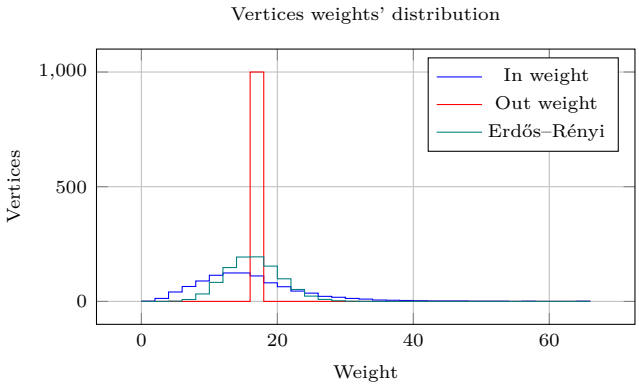
Finally, HAPS periodically trims its BPBA-Tree by replacing each trim leaf with one of the leaves that descend from it. Similar to the above cases, the protocol operates by selecting a TR-leaf and traversing its subtree by taking a random (0.5 probability) step at each node until reaching a leaf. It then prunes the subtree and replaces the TR-leaf by the selected leaf. For example any of the descendants of node $626c/14$, which is both a TR- and an EP-leaf, may replace $626c/14$ after the trim operation. After processing a given TR-leaf, HAPS repeats the process on another one until no TR-leaf has any child nodes left.

IV. Evaluation

We split our evaluation into three parts. We first assess HAPS's ability to protect against Sybil attacks by stating an important property. Then, we analyze its impact on the topology of the resulting overlay graph. Finally, we conclude by evaluating the impact of HAPS on the privacy-preserving averaging protocol presented in [11].



(a) Vertices degrees' distribution for 1000 peers, 16 bits addresses, $l_p = 4$, $l_t = 6$



(b) Vertices weights' distribution for 1000 peers, 16 bits addresses, $l_p = 4$, $l_t = 6$

Figure 2: Vertices degrees' and weights' distribution compared to an Erdős–Rényi random graph

A. Resistance to Sybil Attacks

We start by observing that the use of *equal-probability groups* makes it impossible for an attacker to completely fill a target view with malicious peers as shown by the following Property IV-A.

Property. *Let t be a target peer whose view already contains references to peers in multiple equal-probability groups. HAPS makes it impossible for an attacker to fill t's view with malicious entries unless he/she controls at least a peer in each equal probability group.*

Informal Proof. The proof derives directly from the statement and the properties of EP groups. Since there is at least one group in t's BPBA tree that is not controlled by the attacker, t's view will always contain at least one non-malicious peer. □

B. Impact on the Overlay Graph

Next, we evaluate the impact of HAPS on the overlay graph by simulation. To model the fact that each peer has a specific probability of being sampled, depending on the size of its *equal-probability group*, we represent the overlay graph as a weighted directed graph. An edge from node A to node B with weight $\frac{1}{2^d}$ means that peer B is in the BPBA-Tree of peer A and that peer B is at depth d from its EP-leaf. For example, in Figure 1 peer $626d/16$ is at depth 1 from EP-leaf $626c/14$, and is thus associated with a weight of 0.5. Unless otherwise specified, we consider a network of 1000 peers with 16-bit addresses. In addition, we configure all peers to use $l_p = 4$ and $l_t = 6$. Results are averaged over 10 runs.

We compare the properties of our graph with those of an instance of the oriented variant of Erdős–Rényi random graphs configured with the same number of nodes, and consider two sets of metrics. Weight-unaware metrics do not take edge weights into account and comprise *degree distribution* and *average shortest path length*. Weight-aware metrics take edge weights into account and comprise *weighted degree distribution* and *clustering coefficient*.

For a fair comparison, we configure the Erdős–Rényi graph to match the corresponding HAPS topology in terms of out-degree. When measuring weight-unaware metrics, we set the edge probability in the Erdős–Rényi model so as to obtain an out-degree equal to the number of TR-leaves in the BPBA Tree (2^{l_t}), which represents the size of HAPS's view after each trimming operation. When measuring weight-aware metrics, we instead choose an edge probability such that the expected out-degree of nodes matches the number of EP-leaves in HAPS's BPBA Tree (2^{l_p}), as this better captures the diversity of peer selection provided by HAPS.

1) Weight-Unaware Degree Distribution: Figure 2a shows the in-degree and out-degree distributions resulting from the execution of HAPS. Unlike classical peer-sampling protocols, the out-degree vary across nodes, as different peers may have executed different numbers of pull operations since their latest trimming. The distribution remains nonetheless well centered around its mean. In-degrees exhibit a more spread-out distribution, which results from the distribution of peers on the address plan.

We compare in- and out-degrees with those of the directed Erdős–Rényi variant. Figure 2a shows only one line because for Erdős–Rényi because the two distributions coincide in this case. HAPS exhibits a larger in-/out-degree than Erdős–Rényi because its view grows and shrinks with a minimum size of 2^{l_t} as a result of periodic trimming operations. For the same reason, HAPS's distributions exhibit higher variance.

2) Weighted Degree Distribution: HAPS maintains larger views to implement *equal-probability groups*. However, the diversity it provides in peer selection depends on the number of its EP-leaves. To better reflect this fact, we now weigh each incoming or outgoing edge of the HAPS graph by its associated probability weight. The Erdős–Rényi graph is configured as explained above.

As expected, Figure 2b shows a very thin distribution for weighted out-degrees, centered around the number of EP-leaves; this is a natural consequence of the weighting.

Weighted in-degrees on the other hand vary in a similar manner to those in the Erdős–Rényi baseline, even if the use of *equal-probability groups* makes the distribution a bit more uneven (spread out). This constitutes a side effect of HAPS's Sybil protection.

3) Average Shortest Path Length: Next, we evaluate the average shortest path length of graphs of different sizes for different values of l_p and l_t. Figure 3a shows that very small networks result in complete graphs with an average shortest path length 1. But the value increases with the number of peers. Lower values of l_p and l_t make this increase faster by reducing the number of edges.

In general, we get average shortest path lengths lower than those of Erdős–Rényi random graphs (shown as dashed lines). This can be explain by the fact that peers that are isolated in the address plan end up being present in a large number of local views, becoming effective hubs for the network. In some cases, though, this effect can be overwhelm by the effect of the uneven effective distribution of peers in the address plan, than may cause some local views to be small than they should be after periodic trimming operations. Most importantly, the average shortest path length remains low and comparable to that of random graphs in all configurations.

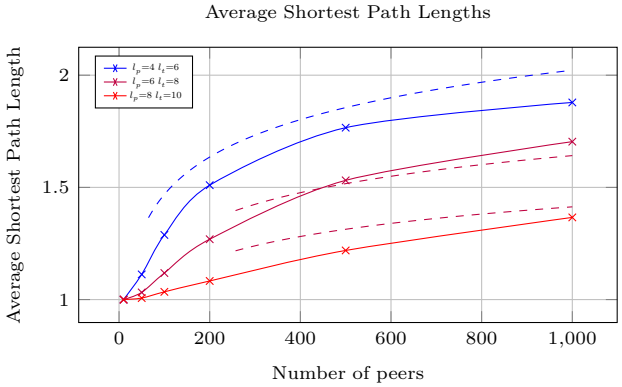
4) Clustering Coefficient: We evaluate the clustering level using the *Clemente-Grassi Clustering Coefficient* defined in [12]. We choose this generalization of the clustering coefficient to directed weighted graph rather than older formulas because of its good properties. Most notably, unlike the one defined in [13], this coefficient is not deflated by low-weight edges. We average results over 10 runs.

Figure 3b shows that very small networks result in complete graphs with a clustering coefficient of 1. But the clustering coefficient lowers as we increase the number of peers. Lower values of l_p and l_t make this decrease faster as this reduces the number of edges. We see that the clustering coefficient is greater than for Erdős–Rényi random graphs. This results from the use of neighbors' views to update a peer's own view.

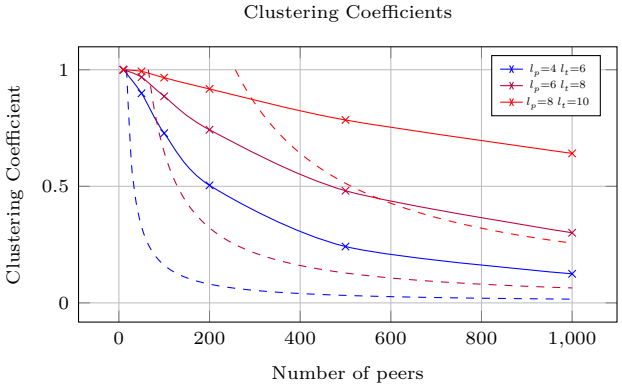
C. Performance for averaging

Finally, we evaluate the convergence speed of the private gossip-based averaging protocol proposed in [11] when running on HAPS (Figure 4b) and on a baseline that uses a request-pull scheme like HAPS but without a Sybil-resistant BPBA Tree (Figure 4a taken from [11]). We first let the respective peer sampling protocols construct their local views. Then we start the averaging process and plot the values of 40 peers for each case as the protocol converges after its privacy-generation phase.[1] As expected, HAPS slightly slows down convergence because its views are less random than the baseline's, but the difference remains limited.

[1]The protocol in [11] operates in two phases. First peers introduce random values in a privacy-generation phase. Then the protocol starts converging.



(a) Average Shortest Path Lengths for various parameters values



(b) Clustering Coefficients for various parameters values

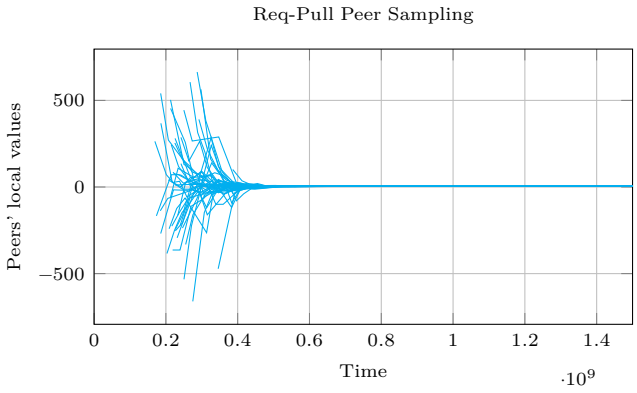
Figure 3: Clustering Coefficients and Average Shortest Path Lengths Erdős–Rényi random graphs (dashed lines)

V. Related work

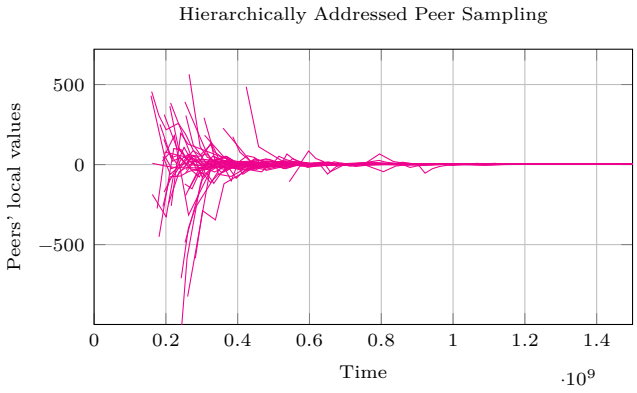
Most work in the context of peer sampling does not consider the problem of malicious behavior and Sybil attacks [14], [7], [8], [15], [16]. Nonetheless, a handful of papers have proposed possible solutions in recent years.

The work in [17], [18] exploits ideas from social-network analysis and uses a knowledge base to store information about peer advertisements and uses it to detect malicious peers that repeatedly advertise colluding peers. However, unlike HAPS, the approach requires the use of cryptographic signatures. Puppetcast [10] handles some byzantine attacks like advertising the identifiers of dead nodes, but it relies on a central authority to address any kind of Sybil attack [3].

Brahms [9] uses the concept of min-wise independent permutation to randomly peers from a possibly biased sample. This allows Brahms to converge to a fixed view despite malicious peers that spam others with the identifiers of other malicious nodes but Brahms also relies on a central authority to counter the use of multiple identifiers



(a) Peer's values convergence with Req-Pull Peer Sampling



(b) Peer's values convergence with Hierarchically Addressed Peer Sampling

Figure 4: Peer's values convergence

for the same node. Moreover Brahms does not provide any refresh mechanisms rather than rerunning the protocol. A recent paper [19] shows how count-min sketches [20] can enable periodic refreshes in a Brahms-like protocol. But it does not solve the need of a central authority for handling the use of multiple identifiers for the same node. HAPS avoids the need of an additional central authority by leveraging the use of IP addresses as identifiers and its BPBA Tree structure.

VI. Conclusion

We proposed a new Sybil-resilient gossip-based peer sampling protocol designed to work in hierarchically-addressed networks. Our approach relies on a novel BPBA Tree, a tree-based container that replaces the classical set typically used in peer-sampling protocols. Our evaluation confirms the effectiveness of our approach in providing peers with a random sample of the network while protecting them from Sybil attacks. Our experiments with a privacy-preserving averaging protocol show that our protocol has only limited impact on performance.

As a future development, it would be interesting to evaluate the performance of our protocol in the context of gossip- and sampling-based Blockchain proposals. In addition, it would be interesting evaluate whether our BPBA Tree data structure can improve other approaches for Byzantine-resilient peer sampling.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP '17. Association for Computing Machinery, 2017, p. 51–68.

[3] J. R. Douceur, "The sybil attack," in *Revised Papers from the First International Workshop on Peer-to-Peer Systems*. Springer-Verlag, 2002, pp. 251–260.

[4] Team Rocket, "Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies," 2018.

[5] D. Frey, A. Mostéfaoui, M. Perrin, P.-L. Roman, and F. Taïani, "Speed for the elite, consistency for the masses: differentiating eventual consistency in large-scale distributed systems," in *Proceedings of the 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS 2016)*, 2016, pp. 197–206.

[6] R. Guerraoui, P. Kuznetsov, M. Monti, M. Pavlovič, and D.-A. Seredinschi, "Scalable byzantine reliable broadcast," in *LIPIcs–Leibniz International Proceedings in Informatics*, D. Publishing, Ed., no. 146. Jukka Suomela, 2019, pp. 22:1–22:16.

[7] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "Gossip-based peer sampling," *TOCS*, vol. 25, no. 3, 2007.

[8] B. Nédelec, J. Tanke, P. Molli, A. Mostéfaoui, and D. Frey, "An adaptive peer-sampling protocol for building networks of browsers," 2017.

[9] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

[10] A. Bakker and M. van Steen, "Puppetcast: A secure peer sampling protocol," in *2008 European Conference on Computer Network Defense*, 2008, pp. 3–10.

[11] A. Bouchra Pilet, D. Frey, and F. Taïani, "Robust privacy-preserving gossip averaging," in *Stabilization, Safety, and Security of Distributed Systems*, ser. Lecture Notes in Computer Science, M. Ghaffari, M. Nesterenko, S. Tixeuil, S. Tucci, and Y. Yamauchi, Eds., vol. 11914. Springer International Publishing, 2019, pp. 38–52.

[12] G. P. Clemente and R. Grassi, "Directed clustering in weighted networks: A new perspective," *Chaos, Solitons & Fractals*, vol. 107, pp. 26–38, 2018.

[13] G. Fagiolo, "Clustering in complex directed networks," *Physical Review E*, vol. 76, p. 026107, 2007.

[14] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kuznetsov, and A.-M. Kermarrec, "Lightweight probabilistic broadcast," *ACM Transactions on Computer Systems (TOCS)*, vol. 21, no. 4, pp. 341–374, 2003.

[15] N. Tölgyesi and M. Jelasity, "Adaptive peer sampling with newscast," in *Euro-Par 2009 Parallel Processing*, ser. Lecture Notes in Computer Science, H. Sip, D. Epema, and H.-X. Lin, Eds., vol. 5704. Springer Berlin Heidelberg, 2009, pp. 523–534.

[16] S. Voulgaris, D. Gavidia, and M. van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and Systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[17] G. P. Jesi, E. Mollona, S. K. Nair, and M. van Steen, "Prestige-based peer sampling service: Interdisciplinary approach to secure gossip," in *Proceedings of the 2009 ACM Symposium on Applied Computing*, ser. SAC '09. ACM, 2009, pp. 1209–1213.

[18] G. P. Jesi, A. Montresor, and M. van Steen, "Secure peer sampling," *Computer Networks*, vol. 54, no. 12, pp. 2086–2098, 2010.

[19] E. Anceaume, Y. Busnel, and B. Sericola, "Uniform node sampling service robust against collusions of malicious nodes," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2013, pp. 1–12.

[20] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, p. 58–75, 2005.